REMARKS

The Specification

The present application for patent is a continuation of 08/957,512 filed on October 24, 1997. The specification, figures, and abstract are copies of that application as originally filed.

A number of minor typographical errors were discovered upon review of the Specification. The above amendment to the Specification corrects those errors. No new matter has been added.

Correction to Drawings:

Enclosed is a replacement sheet for sheet 4 of the drawings with corrections in red. Elements 46, 47, 48, and 49 were previously numbered incorrectly. However, the discussion of these elements on Page 17 of the Specification makes it clear that the intended numbering should be as on the Replacement drawing sheet.

Also enclosed are replacement sheets for Figures 16 and 18 with corrections in red. In both instances the arrow leading into element 166 incorrectly originated with element 165 rather than element 164. Again, the proper flow through the algorithm is evident from the Specification.

The Claims

Claims 1-105 are canceled herein. New Claims 145-188 are added herein. To avoid confusion with the claims in the parent application, Applicant has numbered the claims herein beginning with claim number 145. Claims 145-188 are pending in the application.

Conclusion

Consideration and allowance of Claims 145-188 is respectfully requested.

The additional claim fees have been indicated on the amendment transmittal letter. If Applicant is in error as to these fees, the Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account 19-0597.

Respectfully submitted,

[illegible]

Pehr B. Jansson
Registration No. 35,759

Date: October 23, 2001

Enclosures:

1. Version with Markings to Show Changes Made (3 pages)
2. Submission of Proposed Drawing Amendment (1 page)
3. Figs. 4, 16, & 18 with corrections marked in red ink (3 pages)

Customer No. 26751
Schlumberger Austin Technology Center
Attn: Pehr B. Jansson, Intellectual Property Law Dept.
8311 North FM 620
Austin, TX 78726
Tel: 512-331-3748
Fax: 512-331-3060

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the Specification:

Paragraph beginning at line 18 of page 3 has been amended as follows:

Each kind of memory is suitable for different purposes. Although ROM is the least expensive, it is suitable only for data that is unchanging, such as operating system code. EEPROM is useful for storing data that must be retained when power is removed, but is extremely slow to write. RAM can be written and read at high speed, but is expensive and data in RAM is lost when power is removed. A microprocessor system typically has relatively little ROM and EEPROM, and has 1 to 128 megabytes of RAM, since it is not constrained by what will fit on a single integrated circuit device, and often has access to an external disk memory system that serves as a large writable, non-volatile storage area at a lower cost [that]than EEPROM. However, a microcontroller typically has a small RAM of 0.1 to 2.0 K, 2K to 8K of EEPROM, and 8K – 56K of ROM.

Paragraph beginning at line 3 of page 5 has been amended as follows:

Due to the concern for security, applications written for integrated circuit cards have unique properties. For example, each application typically is identified with a particular owner or identity. Because applications typically are written in a low-level programming language, such as assembly language, the applications are written for a particular type of microcontroller. Due to the nature of low level programming languages, unauthorized applications may access data on the integrated circuit card. Programs written for [a]an integrated circuit card are identified with a particular identity so that if two identities want to perform the same programming function there must be two copies of some portions of the application on the microcontroller of the integrated circuit card.

Paragraph beginning at line 25 of page 6 has been amended as follows:

Among the advantages of the invention are one or more of the following. New applications may be downloaded to a smart card without compromising the security of the smart card. These applications may be provided by different companies loaded at different times using different terminals. Security is not [comprised]compromised since the applications are protected against unauthorized access of any application code or data by the security features provided by the Java virtual machine. Smart card applications can be created in high level languages such as Java and Eiffel, using powerful mainstream program development tools. New applications can be quickly prototyped and downloaded to a smart card in a matter of hours without resorting to soft masks. Embedded systems using microcontrollers can also gain many of these advantages for downloading new applications, high level program development, and rapid prototyping by making use of this invention.

Paragraph beginning at line 24 of page 15 has been amended as follows:

The terminal 14 can also interact with applications running in the integrated circuit card 10. In some cases, different terminals may be used for these purposes. For example, one kind of terminal may be used to prepare applications, different terminals could be used to download the applications, and yet other terminals could be used to run the various applications. Terminals can be automated teller machines [(ATM)s] (ATMs), point-of-sale terminals, door security systems, toll payment systems, access control systems, or any other system that communicates with an integrated circuit card or microcontroller.

Paragraph beginning at line 6 of page 18 has been amended as follows:

To avoid dynamic linking in the card, all the information that is distributed across several Java class [file]files 24a, 24b, and 24c that form the application 24, are coalesced into one card class file 27 by the process shown in the flowchart in Fig. 5. The first class file to be processed is selected 51a. The constant pool 42 is compacted 51b in the following manner. All objects, classes, fields, methods referenced in a Java class file 24a are identified by using strings in the constant pool 42 of the class file 24a. The card class file converter 26 compacts the constant pool 42 found in the Java class file 24a into an optimized version. This compaction is achieved by mapping all the strings found in the class file constant pool 42 into integers (the size of which is microcontroller architecture dependent). These integers are also referred to as IDs. Each ID uniquely identifies a particular object, class, field or method in the application 20. Therefore, the card class file converter 26 replaces the strings in the Java class file constant pool 42 with its corresponding unique ID. Appendix B shows an example application HelloSmartCard.java, with a table below illustrating the IDs corresponding to the strings found in the constant pool of the class file for this application. The IDs used for this example are 16-but unsigned integers.

In the Claims:

Claims 1-105 have been cancelled. New Claims 145-188 have been added herein.

To avoid confusion with the claims in the parent application, Applicant has numbered the New Claims beginning with claim number 145. Claims 145-188 are pending in the application.